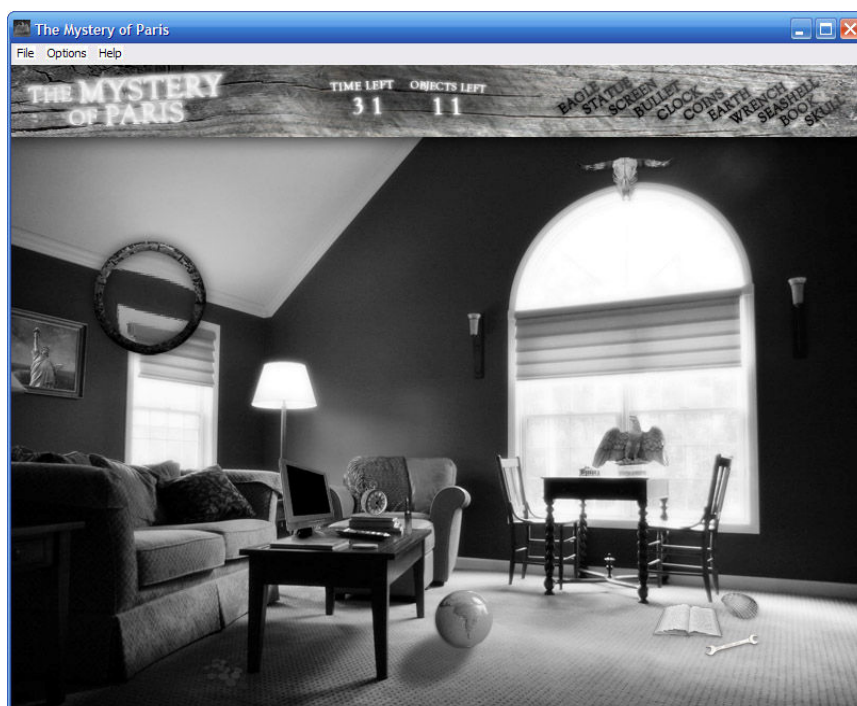


# *the* **MYSTERY** *of Paris*

You may not use this tutorial for any other purpose than learning, working or having fun... In other words: You can use this PDF tutorial for anything you'd like, as long as it doesn't involve both a hammer and a squirrel.

**Koobare**  
marchewkowy@gmail.com

**Welcome** to another one of **Koobare's little tutorials**, teaching you – as always – how to effectively and efficiently use the best multimedia authoring tool ever – [Multimedia Fusion 2](#) by Clickteam! This tutorial is meant for beginner intermediates, people who already played with a couple of projects in MMF2 and know their way around the interface and Fusion's basic features. Take a look at this simple lesson guide to help you decide which tutorials should be played with first and which should be left for later (note that this is just a suggestion):

<b>MMF2 Interface:</b> Interface Guide + Image Editor Guide	<i>First time with MMF2</i>
<b>Basics:</b> Smelly Claw tutorial	<i>Beginners</i>
<b>Game tutorials:</b> Glob Wars and/or Risky Waters	<i>Beginners</i>
<b>Game tutorials:</b> You've Got Spacemail!	<i>Beginner-intermediates</i>
<b>You are here → The Mystery of Paris</b>	<i>Beginner-intermediates</i>
<b>Next:</b> Castle Defender and/or Space Corsair	<i>Beginner-intermediates</i>
<b>Next:</b> Save & Load tutorial	<i>Intermediates</i>

In this tutorial we will create a nice-looking “Hidden Object” game, in which the player has to locate items concealed somewhere on the screen. This type of games – known also as the “Seek and Find” or “Screen Detective” games – have become the hottest casual playthings during the last few years, with a bunch of game design companies releasing literally hundreds of such titles every year.

### Our story begins...

**Johnny MacCullagan** was a homicide detective for the Los Angeles Police Department – one of the best in his line of job, even if a bit hot-tempered. He left no crime unpunished, no murderer ever walked the streets again after Johnny brought him to interrogation and dived deep into evidence. A fatal accident at the precinct – in the course of which he lost his left ear – unfortunately cost him his job... But Johnny decided to carry on, **opening a Private Eye agency**, promising to solve any case for a nominal fee.

For three years he bounced from one divorce case to another, spying on wives and husbands alike, devoid of the thrill that kept him going during his LAPD years. He began to feel bored, gained some weight and even thought about closing down the business to open up a nice bar or sumthin', when SHE appeared at his door, asking for help.



Her curls were red like gas-fueled flames, eyes glittered in the noontime sun, her voice sounded like that of an angel... She introduced herself as **Gabriella Paris, a true damsel in distress**, looking for someone who could find her kidnapped brother. **Joshua Paris**, who traveled to the capital of France three years ago, went missing last week. His New York apartment was ransacked, rummaged through by someone looking for something in a real hurry. Johnny gave Gabriella a reassuring roguish smile and accepted the assignment with the words he always wanted to say out loud: "I'm on it, like ants on a piece of blueberry pie".

It didn't take him long to arrive at Paris. Not knowing even a word in French -- aside from "merci", that is -- he felt as alienated as one could be, not even knowing where to go to rent a room or how to order a cup of milk. He decided to focus on the task at hand, leaving sight-seeing for later -- Gabriella wrote him Joshua's work address on a small piece of paper that he kept in his pocket. He gave the little note to a taxi driver and soon arrived at the doors of the *Lewis, Pegg & Paris Smirking Eagle Company*, a mysterious import-export corporation specializing in antiques and art, which actually wasn't registered in France, or even in Europe -- the records showed that the company didn't even have an European address, since it was founded and based in South America. It seemed that no one was at the office, with the doors locked down with a chain and a padlock.

Johnny wasn't one of those types that would sit around and wait for someone to arrive, while poor Joshua was out there, most probably kept captive, perhaps even at gunpoint. With the help of a set of lockpicks that he always carried around with him, he got inside the building, traversing its dark, long corridors. Something wasn't right here -- Johnny had a hunch that this

case was going to turn into something ugly and really fast. Everywhere he looked, there was a disturbing image of a smiling eagle – the main hall was full of bird sculptures, some of them placed on knee-high columns, with the walls painted in such a way, that you could see a pair of winged creatures everywhere you looked.

He finally found Joshua's room, ready to collect some random samples and pieces of evidence that could help him decipher this bizarre case. He was going through Joshua's books – mostly about uncommon American birds, forgotten ancient rituals and Aztec mythology – when he



heard a familiar sound, fathoming in the distance... Police sirens! Someone called the police, trying to frame him into whatever happened here and Johnny was pretty sure that they would be storming in at any time now. All he had was sixty seconds – no more than a minute to gather everything that could be useful to unravel the mystery of Joshua's disappearing.

"I need to take some random stuff to check out fingerprints", he instructed himself. There was a book about eagles, a globe-shaped ball, a wrench and a bunch of other stuff laying around. "All of this could be useful!", Johnny said to himself, as the sirens became louder and louder. There was no more time... He had to grab everything and get out as fast as possible...

### What we're going to create...

So, you now know the story behind our little game, **now it's time to lay out what exactly are we trying to achieve here...** The basic concept goes like this: the player has exactly 60 seconds (measured down by a counter) to gather up 11 objects thrown around the room. He does so by clicking on them with his left mouse button – if he clicks on an object, it disappears from the room and it's name is scratched out from the list that can be found on top of the screen. Some items are named in such a way that the player has to guess what exactly does the name from the list represent (for example: "Earth" means a globe-shaped ball, whereas "Statue" is a cut-out cardboard Statue of Liberty placed on a painting).

To add a nice graphical effect to our little game, we'll use a **Lens object**, set to standard, unaltered preferences – when it will be combined with another object (just a simple active object that will be positioned on top of the lens), it will resemble the well-known detective tool used by the world's most famous investigators, from Sherlock Holmes to Mr. Hercules Poirot – a magnifying glass. Trust me – with just two or three lines of code, this will add a really nice touch to our project.

And – talking about nice touches – let's not forget about other **graphical improvements** that we're going to introduce into our game. First of all, every time the player clicks on one of those eleven clickable items, they will disappear in a blast of light, exploding and fading out of the screen. The lines that will be used to "scratch out" the items from the to-do list will use a fade-in, which will make them look animated, even though they only have a single frame of animation. The third and last thing – we will use a semi-transparent object (it's transparency will be image-based, we won't use MMF2's native Ink Effects this time) with a simple jiggle animation to cover the title of our game and make it a bit more attention-grabbing.

Two more things to add here: a word about counters and a second one about sound. Yeah, there will be more counters than just the timer one – another one will be used for counting down how many pickupable items are there still in play, and a third one will be used to delay the final jump to another frame, after the player singled out all the required stuff. As for the sound: we'll use the sounds and music provided on MMF2's Bonus disc to enhance the overall feel of the game. You'll see what I mean in a few minutes – let's carry on, detectives...

- ☐ **If you have any problems with this tutorial, or notice that there are some mistakes present, please, contact me and I'll do my best to help you and replace all the errors with correct information.**

Contact me at: [marchewkowy@gmail.com](mailto:marchewkowy@gmail.com)

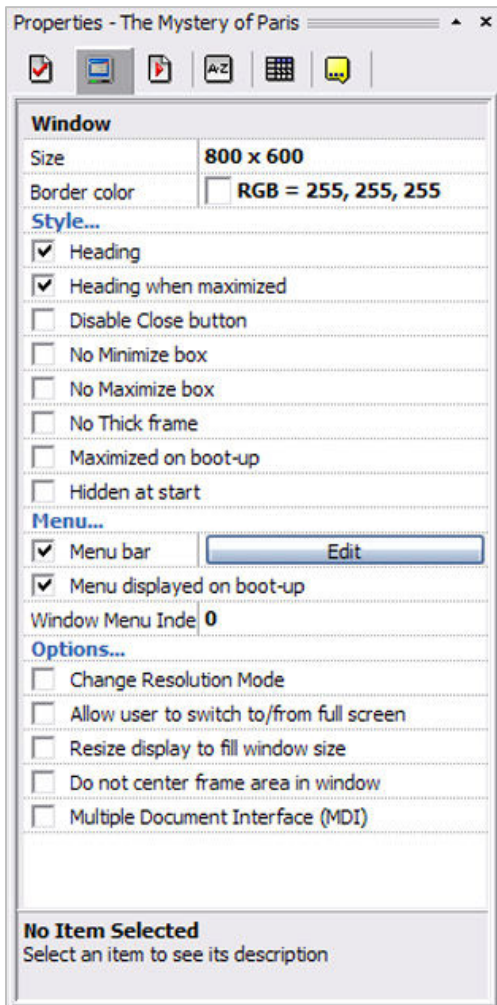
- ☐ **Note: I've been receiving some reports that not all e-mails get to me for some reason. Seems that some of them (quite a lot) end up in my spambox or are blocked out by the server.** I dunno why this is happening, so if you're experiencing any difficulties with delivering me a message or haven't received a reply in quite some time, please, send me another e-mail at [marchewkowy@wp.pl](mailto:marchewkowy@wp.pl), making sure that its title begins with "To Koobare:". I'll do my best to check both these e-mails regularly.



## Part I: Setting up the application.

---

As we usually do, we'll start off simple, with the most basic task all MMF2 users have to do from time to time – in other words, I'll show you how to create and set up our app. If you already know how to do this and are worn-out by such a basic approach, just head to **Part II** of this tutorial (just scroll down). This section is meant for the newbies, so you don't have to waste your time here if you're more of a beginner-intermediate user.



Anyways, let's do this... Open Multimedia Fusion 2, **create a new application** and save it onto your drive (remember that it's always a neat idea to have the *Autobackup* option of MMF2 turned on – check your MMF2's *Preferences*). Rename it to “**The Mystery of Paris**”. Now, go to your application's **Properties window** (if it didn't open up by itself, right click on your application's name in the workspace toolbar and select *Properties* from the drop-down menu), and select the **Window** tab (second from the left). Set the window size to **800x600** (it's one of the default sizes, so you can select it from the quick-selection drop-down). If MMF2 asks you if you'd like to modify the size of the existing frame as well, select “Yes”. If it doesn't ask you by itself, open up the Storyboard Editor and change the size of the frame to **800x600**. Rename the frame to “The Mystery of Paris” (if you're planning on adding some of your own levels later on, rename it to something easily systematizable, for example “Level 1”).

📄 **Interested in one game type in particular? Would like to learn about something that hasn't been covered in any of the released tutorials yet? Got an idea that could interest other tutorial-readers? Just drop me an e-mail!**

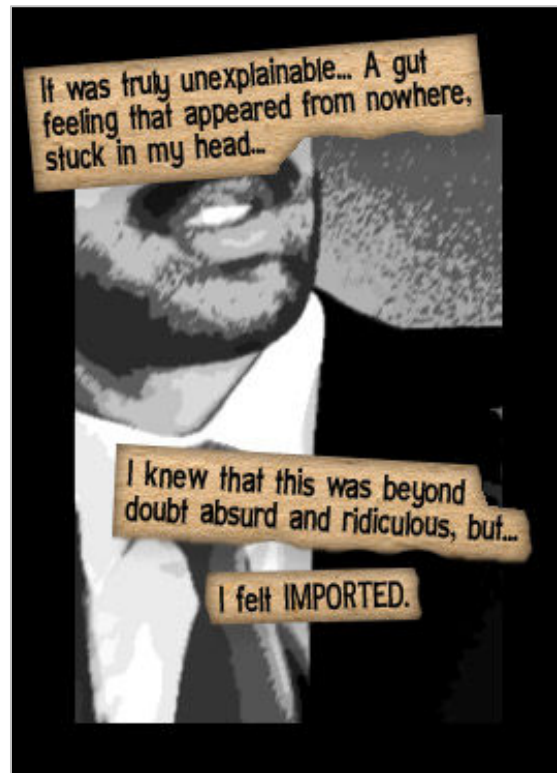
Contact me at: [marchewkowy@gmail.com](mailto:marchewkowy@gmail.com) or [marchewkowy@wp.pl](mailto:marchewkowy@wp.pl)

## Part II: Making your life a bit easier...

---

In one of my previously created tutorials – the “You’ve got Spacemail!” one – I’ve explained in detail why I’ve decided that in part two of every *beginner-intermediate* and *intermediate* level tutorials we will dump everything we created in the optional part one of the tut (which was just for the sake of showing you how it’s done) and open up a pre-prepped app, which will be used as a base for our project.

Let’s not waste time on organizational issues, since you can find the reasoning behind this in the previously mentioned Spacemail-tut... To continue, just close what you’ve done so far (if you actually didn’t just skip section one) and open the **mystery-of-paris-basis.mfa** file.



If you’re wondering where can this file be located, just take a look at the directory where you’ve unpacked the PDF tutorial that you’re currently reading – both these files were zipped into the same archive, so you’ve most probably unpacked them to the same file folder.

**Please note:** most of the objects used in this tutorial use alpha channels, a feature that is unfortunately unavailable in Games Factory 2 (TGF2 users should use basic library objects or create their own graphics instead – I generally would advise you to upgrade to MMF2 as soon as possible, since you are missing out on some really good stuff, a lot of quite impressive and useful features and additional objects).

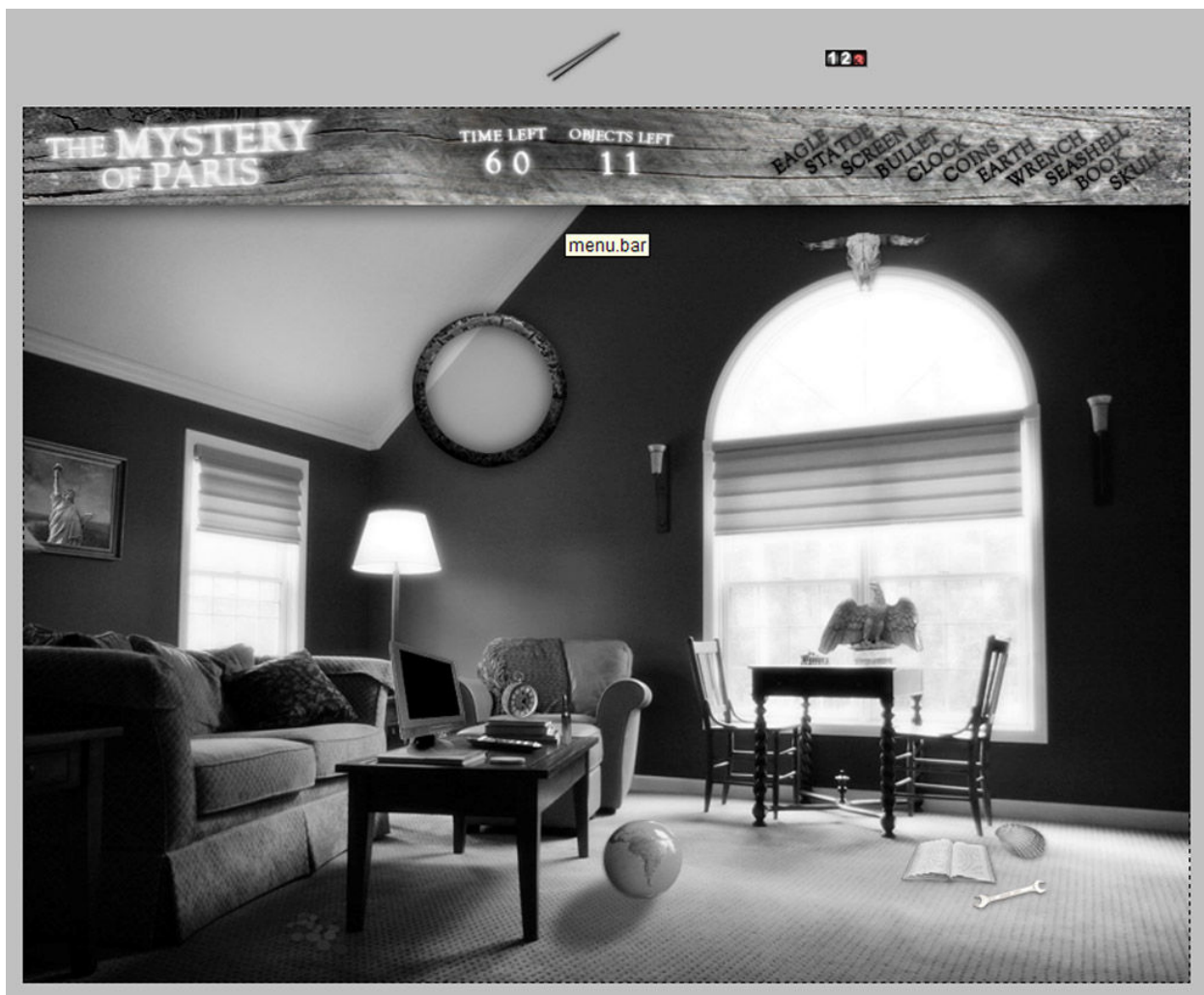
**Full game:** the finished tutorial game is included in the same archive.

### Part III: Investigation.

---

Once you've opened our basis application, the foundation on which we're going to build our game, go right into the first frame and take a look around (for those of you who relish for a bit more drama: try to do it in slow-motion, while flying through the air, Max Payne-style). There's a whole gang of objects laying around, looking really pretty, just waiting for you to script them all together into a nice, simple, "Hidden Object" game.

Your Frame Editor should look somewhat like this right now:



Notice that all objects have been conveniently placed to their proper positions... Having said that, there's no problem with moving them around – you may move the Earth-shaped ball more to the right or drop the monitor screen on the floor, and the game still will work properly. Anyways, let's leave all these objects as they are right now and move on, I'm sure you can't wait to dive into the programming part of this tut...



Yet, before we can move on to the scripting, we should firstly learn all there is to know about the objects used in our application – what will they be used for, which ones have a fade-in transition, what are their basic preferences... That kind of stuff. Below you can find an alphabetical list of all the objects, with their short description:

Object's name:	So... What is it?
<b>backdrop.photo</b>	<p><u><i>The background for our game.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>In most games I would describe this thingie as "just a background object", but in this particular game it's an essential part of the gaming experience...</i></p> <p><i>First of all, this is basically <b>the main thing the player sees throughout the whole level</b>... There are no jumping spacemen, hovering helicopters, dashing sports cars and other animated stuff rushing through the screen... In other words: no distractions.</i></p> <p><i>Secondly, the quality of the main backdrop and its details are basically responsible for the <b>difficulty of our game</b>: the more details, the more objects laying around, the more difficult it will be to find the right items before the time runs out.</i></p> <p><i>The backdrop of our game is basically a still image of <b>Joshua's office interior</b>: there are some chairs, a couch, a table, couple of lamps, two windows... Basically: there would be nothing special about this place if it wasn't for the slightly overcontrasted black &amp; white style in which the apartment is presented, adding a strange feeling of an unknown menace or at least a "classic story"-feel, similar to the art style of the Bogart-era (just think about "Casablanca" or "The Maltese Falcon").</i></p>
<b>counter.hidden</b>	<p><u><i>A hidden counter used for controlling our game and its frames.</i></u></p> <p>PROPERTIES</p> <p><i>It is set as a <b>hidden counter</b>. Its <b>Initial and Maximum Values</b> are both set to <b>3</b>, its <b>Minimum Value</b> is set to <b>0</b>.</i></p> <p>WHY IT'S HERE?</p> <p><i>Because we need a simple solution for delaying the frame-jumping event once all the objects are found.</i></p> <p>HOW DOES IT WORK?</p> <p><i>Well, it's a pretty simple system, really... Once all the required items have been picked up by the player (and, thus, the objects counter has a value of zero), this counter looses 1 in value with every second that passes.</i></p> <p><i>You don't have to be a mathematical genius to find out that this will give us a 3-second interlude once all the objects have been singled out (since its Initial Value is set to 3).</i></p>

	<p><i>This will give us just the right amount of time to play a nice victory-related sample and then – once this counter is down to zero – jump to the next frame of our application. Please note that I've used a counter for this just to make things easier and clearer – you could use the Timer object or a Global / Alterable values as well.</i></p>
<b>counter.objects</b>	<p><u>Another counter, this one counts how many items are left in the frame.</u></p> <p>PROPERTIES</p> <p><i>It is set as a <b>Numbers counter</b> (with the images imported from external files). Its <b>Initial and Maximum Values</b> are both set to <b>11</b>, its <b>Minimum Value</b> is set to <b>0</b>.</i></p> <p>WHY IT'S HERE?</p> <p><i>Because it's simpler to place this counter in the frame than to check on every loop how many pickupable items are left in the frame. Furthermore, this also has an informative function, as the player can use it to check how many more items does he still needs to locate.</i></p> <p>HOW DOES IT WORK?</p> <p><i>It starts with its current value set to 11, which is then decreased by 1 every time the player picks up one of the scattered items. Once it's down to 0, the time countdown stops and the hidden counter countdown begins (giving the player three seconds to say goodbye to the level before being transported to the next one).</i></p>
<b>counter.time</b>	<p><u>Our timer – a counter that counts down time.</u></p> <p>PROPERTIES</p> <p><i>It is set as a <b>Numbers counter</b> (with the images imported from external files). Its <b>Initial and Maximum Values</b> are both set to <b>60</b>, its <b>Minimum Value</b> is set to <b>0</b>.</i></p> <p>WHY IT'S HERE?</p> <p><i>Because we need to have a way of telling how many seconds have passed since the player started searching for all those concealed items... Or rather how many time does he have left before the police storms into the room and he officially loses.</i></p> <p>HOW DOES IT WORK?</p> <p><i>It is initially set to 60, which is decreased by 1 on every second that passes (only if there are still items to be picked up – if the player has already clicked them all, the time counter is stopped).</i></p>
<b>Lens</b>	<p><u>The Lens object – used for our little magnifying glass.</u></p> <p>PROPERTIES</p> <p><i>All properties are set to <b>default</b> (which means that the Initial lens effect is set to Zoom, the Multiplier is set to 1.0 and the Autoupdate option is ON).</i></p>

	<p><b>WHY IT'S HERE?</b></p> <p><i>Because we want a nice-lookin' magnifying glass for our Private Eye.</i></p> <p><b>HOW DOES IT WORK?</b></p> <p><i>The Lens object, created by Anders Riggelsen, is basically used to distort everything placed beneath the lens object like if it was seen through a piece of glass, or, well, a lens – this effect is controlled by the usage of a grayscale gradient object, which can be easily edited or imported from outside of MMF2. We use the basic, default "grayscale circle" object, since it perfectly suits our needs.</i></p>
--	--

<b>magnifying.glass</b>	<p><u><i>A metal circle (active object), used to create our magnifying glass.</i></u></p> <p><b>WHY IT'S HERE?</b></p> <p><i>Because our magnifying glass cannot be made of a single lens, can it? Surely not – it needs a nice metal-lookin' border!</i></p> <p><b>HOW DOES IT WORK?</b></p> <p><i>Both the Lens object and this one are always positioned at the player's mouse coordinates (always displayed in the same place as the player's cursor). Thanks to this – and to the fact that this object is always placed on top of the Lens – we are able to secure a nice magnifying glass that follows the mouse cursor on real time.</i></p>
-------------------------	--



<b>menu.bar</b>	<p><u><i>The name says it all – our menu bar (a backdrop object).</i></u></p> <p><b>WHY IT'S HERE?</b></p> <p><i>Because we could really use a menu bar, I guess. It's a backdrop for our counters and it contains the list of pickupable items.</i></p>
-----------------	--

<b>object.book</b> and <b>object.bullet</b> and <b>object.clock</b> and <b>object.coins</b> and <b>object.eagle</b> and <b>object.earth</b> and <b>object.screen</b> and <b>object.seashell</b> and <b>object.skull</b> and <b>object.statue</b> and <b>object.wrench</b>	<p><u>Our pickupable items – the stuff the player needs to find.</u></p> <p>WHY IT'S HERE?</p> <p>Well, this game is all about picking up strange items, so it's a pretty guaranteed thing that there should be such in our frame, right? To be exact: there are eleven of them, check the menu.bar item to find the whole list (or just take a look to the left).</p> <p>HOW DOES IT WORK?</p> <p>Player clicks them, they are destroyed, another line is scratched out from the items list... And that's it.</p> <p>TRANSITIONS</p> <p>The <b>fade-out transition</b> of all these objects is set to "<b>Fade</b>" (2.3 seconds). Thanks to this they fade away into air after being picked up by the player (and, oh, there's also a bright explosion, but we'll get to that part in a few seconds).</p>
<b>scratched.out</b>	<p><u>Black lines that are "scratching out" the items from our to-do list.</u></p> <p>WHY IT'S HERE?</p> <p>To help us show which objects have already been found, by covering up their names on the menu bar list.</p> <p>HOW DOES IT WORK?</p> <p>It's as simple as they get: once an item is located by the player, he clicks on it, it disappears, and this little thingie is created on top of the name representing the clicked item.</p> <p>TRANSITIONS</p> <p>To make it appear as if someone really "scratches out" the names from the list, I've added a simple fade-in transition to this object. The <b>fade-in transition</b> is set to "<b>Bands</b>" (0.67 of a second, coming from left to right).</p>
<b>text.objects</b>	<p><u>Just a small alpha-blended Backdrop object.</u></p> <p>WHY IT'S HERE?</p> <p>To help us easily distinguish the items counter from the time counter – it says "objects left" after all...</p>
<b>text.time</b>	<p><u>Another little Backdrop object with some text on it.</u></p> <p>WHY IT'S HERE?</p> <p>It does the same thing as the previously characterized object, but this one says "time left" and is placed over the time counter.</p>
<b>title.flash</b>	<p><u>A small animation to make the title look better.</u></p> <p>WHY IT'S HERE?</p> <p>A semi-transparent jiggling animation that is placed on top of the game's title (on the menu bar) – check out it's simple animation.</p>

<b>white.blast</b>	<p><u><i>A nice-lookin' white explosion.</i></u></p> <p><b>WHY IT'S HERE?</b></p> <p><i>To make the game look better – once an item is clicked, it doesn't just disappear into thin air, it actually explodes with a great-looking burst of light.</i></p> <p><b>HOW DOES IT WORK?</b></p> <p><i>This little animated object is created on top of an item that has just been clicked by the player. Once it's animation is over, it is destroyed, disappearing from the frame.</i></p> <p><b>TRANSITIONS</b></p> <p>The <b>fade-out transition</b> is set to <b>"Zoom"</b> (0.22 of a second).</p>
--------------------	--

## A word about alpha-channeled objects

As a developer greatly concerned about the audio-visual side of my projects, I tend to use a lot of alpha-channeled objects – actually, I'm not that into pixel art, so I use them every time I create something, always playing with transparencies and blending. The thing that I wanted to emphasize here is that this game wouldn't look so good if it wasn't for alpha-channels, helping us to make it harder to distinguish between a separate object and the background of the game.

## Part IV: It's in the code.

---

It's now time for the scripting part – my personal favorite! Save your project (always remember to save it from time to time!) and open up the **Event Editor**. If you're new to my tutorials, let me introduce you to the event-recording system that I use. If you know it already – just skip this frame below and quickly move on to the coding part:

### *Koobare's MMF-to-paper coding system*

Basically, it goes like this:

**IF (Condition):** [Object for the condition] > Condition group > Condition

**THEN (Action):** [Object for the action] > Action group > Action

Seems simple, right? Well, that's just because IT IS simple. All the conditions are marked in red, while actions are written in fancy blue.



Object names are always put in [square brackets]. The final condition/action is always in *Italic*. If we'll have a multi-condition event, then it'll be like this:

IF (Condition 1): [Object for condition 1] > Condition group 1 > *Condition 1*

IF (Condition 2): [Object for condition 2] > Condition group 2 > *Condition 2*

THEN (Action): [Object for the action] > Action group > *Action*

Whereas a multi-action event looks like this:

IF (Condition): [Object for condition] > Condition group > *Condition*

THEN (Action 1): [Object for the action 1] > Action group 1 > *Action 1*

THEN (Action 2): [Object for the action 2] > Action group 2 > *Action 2*

If you'll have to input anything by keyboard, it will be indicated by coloring the text green and using < angle brackets >, like this (this marking will be soon obsolete):

< Set the Global Value A to 32 >

Additional comments, instructions and info will be put in << double angle brackets >>, using a different color (this marking will soon be also used for input):

<< Select any wave sound from the MMF2's sound library >>

From time to time I'll also use this style to throw in some extra tips and tricks about MMF2 and more advanced coding techniques. All you have to do is to go step-by-step through all the listed events and keep one eye on your Event Editor, and the second one on this tutorial...

## Let's unravel this mystery!

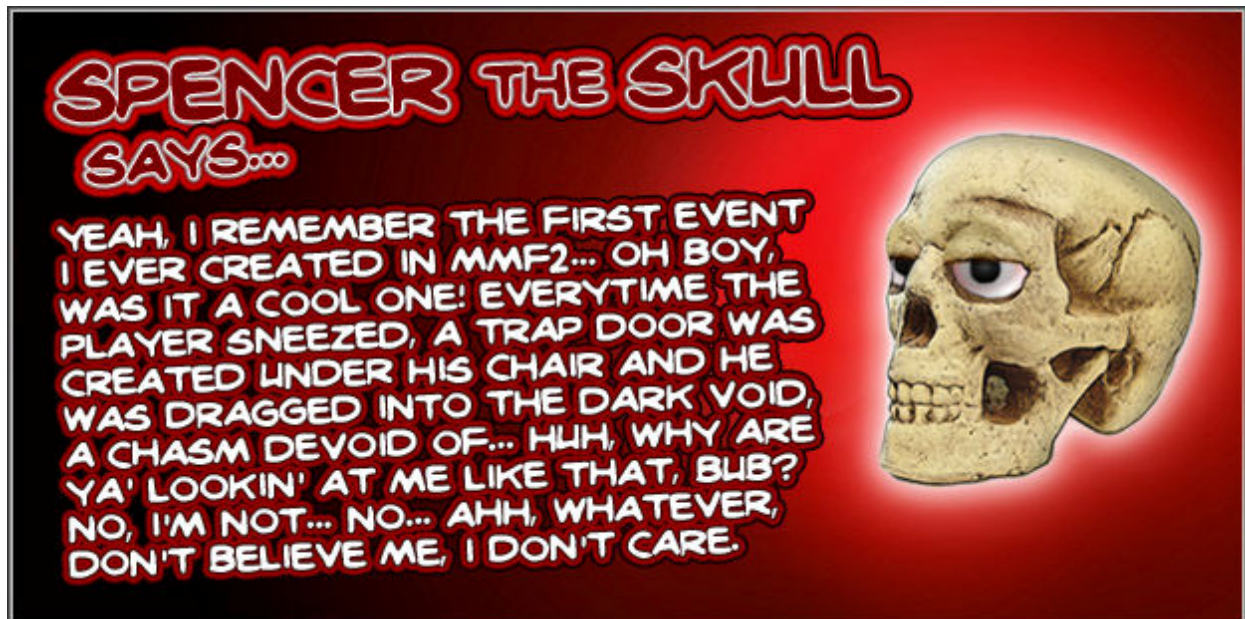
1) Firstly, let's start off with the conventional **"Start of frame"** event, which I usually create at the very beginning of the events list (seems pretty logical, right?). This event – triggered when someone starts our game – will initiate our music track, nothing more, nothing less. I've found a perfect little piece of music on MMF2's Bonus disc, in the \Samples\Music\Magic & Drama folder, but you can of course choose something different if you wish:

**IF:** [Storyboard Controls] > Start of frame

**THEN:** [Sound Object] > Samples > Play and loop sample

<< Choose the "Mysterious Dungeon.wav" sound, loop it 0 times, which means infinitely >>

Got it? Great! That means that we've got our first event up and ready, even if it's a very petite one. Never underestimate the meaning of a game's soundtrack, though...



2) Time for our second event! This one will be as simple as our first one, and will basically play a little sample every time the player clicks with his left mouse button:

**IF:** [Keyboard & Mouse Object] >> The Mouse >> User clicks

<< select: LEFT BUTTON, SINGLE CLICK >>

**THEN:** [Sound Object] >> Samples >> Play sample

<< Choose the "Drip 4.wav" sound, from \Samples\Impacts >>

3) It's now time for a series of more "serious" events, establishing what will exactly happen if the player clicks on one of the pickupable items:

**IF:** [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object

<< select: LEFT BUTTON, SINGLE CLICK >>

<< choose the [object.statue] object >>

**THEN:** [object.statue] >> Destroy

**THEN:** [Create New Objects] >> Create Object

<< Select the [scratched.out] object >>

<< Set the coordinates to x=558, y=30 >>

```

THEN: [Create New Objects] >> Create Object
<< Select the [white.blast] object >>
<< Set the coordinates to x=-5, y=9, relative to [object.statue] object >>
THEN: [counter.objects] >> Subtract from Counter
<< input: 1 >>
THEN: [Sound Object] >> Samples >> Play sample
<< Choose the "VIOLIN1.wav" sound, from \Samples\Music\Jingles >>

```

Let's take a closer look at this event: when the player clicks on the Statue of Liberty cut-out, it is quickly destroyed, it's name being scratched out from the item list (the "scratched.out" object is created on top of it), a white blast is created on top of the vanishing item, the item counter's value is lowered by 1 and a mysterious violin sample is played in the background... Everything's clear, right? Now, let's do the same for the skull hovering above the window:

```

IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object
<< select: LEFT BUTTON, SINGLE CLICK >>
<< choose the [object.skull] object >>
THEN: [object.skull] >> Destroy
THEN: [Create New Objects] >> Create Object
<< Select the [scratched.out] object >>
<< Set the coordinates to x=763, y=43 >>
THEN: [Create New Objects] >> Create Object
<< Select the [white.blast] object >>
<< Set the coordinates to x=0, y=5, relative to [object.skull] object >>
THEN: [counter.objects] >> Subtract from Counter
<< input: 1 >>
THEN: [Sound Object] >> Samples >> Play sample
<< Choose the "VIOLIN2.wav" sound, from \Samples\Music\Jingles >>

```

And here's one for the eagle statue:

```

IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object
<< select: LEFT BUTTON, SINGLE CLICK >>
<< choose the [object.eagle] object >>
THEN: [object.eagle] >> Destroy
THEN: [Create New Objects] >> Create Object
<< Select the [scratched.out] object >>
<< Set the coordinates to x=531, y=31 >>
THEN: [Create New Objects] >> Create Object

```

```

<< Select the [white.blast] object >>
<< Set the coordinates to x=1, y=-1, relative to [object.eagle] object >>
THEN: [counter.objects] >> Subtract from Counter
<< input: 1 >>
THEN: [Sound Object] >> Samples >> Play sample
<< Choose the "VIOLIN3.wav" sound, from \Samples\Music\Jingles >>

```

Here's the computer screen object...

```

IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object
<< select: LEFT BUTTON, SINGLE CLICK >>
<< choose the [object.screen] object >>
THEN: [object.screen] >> Destroy
THEN: [Create New Objects] >> Create Object
<< Select the [scratched.out] object >>
<< Set the coordinates to x=583, y=31 >>
THEN: [Create New Objects] >> Create Object
<< Select the [white.blast] object >>
<< Set the coordinates to x=1, y=-4, relative to [object.screen] object >>
THEN: [counter.objects] >> Subtract from Counter
<< input: 1 >>
THEN: [Sound Object] >> Samples >> Play sample
<< Choose the "VIOLIN4.wav" sound, from \Samples\Music\Jingles >>

```

Aaaand here's one for the alarm clock:

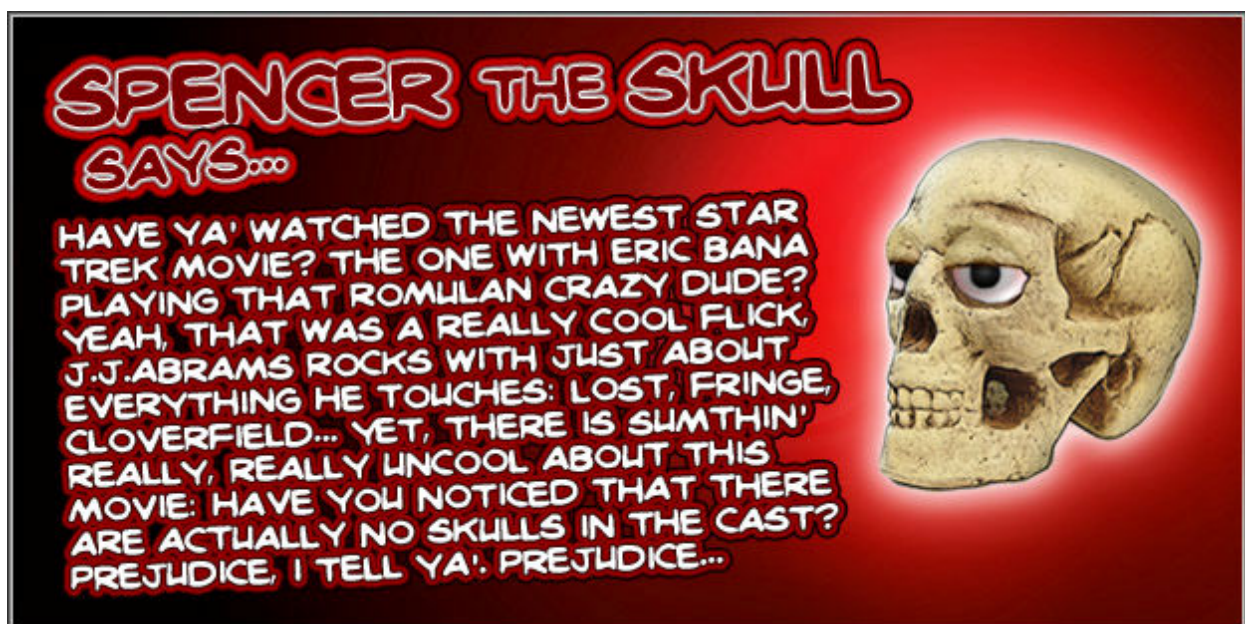
```

IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object
<< select: LEFT BUTTON, SINGLE CLICK >>
<< choose the [object.clock] object >>
THEN: [object.clock] >> Destroy
THEN: [Create New Objects] >> Create Object
<< Select the [scratched.out] object >>
<< Set the coordinates to x=623, y=37 >>
THEN: [Create New Objects] >> Create Object
<< Select the [white.blast] object >>
<< Set the coordinates to x=1, y=2, relative to [object.clock] object >>
THEN: [counter.objects] >> Subtract from Counter
<< input: 1 >>
THEN: [Sound Object] >> Samples >> Play sample
<< Choose the "VIOLIN5.wav" sound, from \Samples\Music\Jingles >>

```

Another event, this time concerning the bullet left on the table:

IF: [Keyboard & Mouse Object] >> The Mouse >> *User clicks on an object*  
<< select: LEFT BUTTON, SINGLE CLICK >>  
<< choose the [object.bullet] object >>  
THEN: [object.bullet] >> *Destroy*  
THEN: [Create New Objects] >> *Create Object*  
<< Select the [scratched.out] object >>  
<< Set the coordinates to x=604, y=32 >>  
THEN: [Create New Objects] >> *Create Object*  
<< Select the [white.blast] object >>  
<< Set the coordinates to x=0, y=0, relative to [object.bullet] object >>  
THEN: [counter.objects] >> *Subtract from Counter*  
<< input: 1 >>  
THEN: [Sound Object] >> Samples >> *Play sample*  
<< Choose the "VIOLIN6.wav" sound, from \Samples\Music\Jingles >>



Once again a very similar event, this one is for the seashell laying on the floor:

IF: [Keyboard & Mouse Object] >> The Mouse >> *User clicks on an object*  
<< select: LEFT BUTTON, SINGLE CLICK >>  
<< choose the [object.seashell] object >>  
THEN: [object.seashell] >> *Destroy*  
THEN: [Create New Objects] >> *Create Object*  
<< Select the [scratched.out] object >>



```

<< Set the coordinates to x=727, y=35 >>
THEN: [Create New Objects] >> Create Object
<< Select the [white.blast] object >>
<< Set the coordinates to x=-1, y=-2, relative to [object.seashell] object >>
THEN: [counter.objects] >> Subtract from Counter
<< input: 1 >>
THEN: [Sound Object] >> Samples >> Play sample
<< Choose the "VIOLIN7.wav" sound, from \Samples\Music\Jingles >>

```

And here goes the book:

```

IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object
<< select: LEFT BUTTON, SINGLE CLICK >>
<< choose the [object.book] object >>
THEN: [object.book] >> Destroy
THEN: [Create New Objects] >> Create Object
<< Select the [scratched.out] object >>
<< Set the coordinates to x=740, y=41 >>
THEN: [Create New Objects] >> Create Object
<< Select the [white.blast] object >>
<< Set the coordinates to x=-1, y=-4, relative to [object.book] object >>
THEN: [counter.objects] >> Subtract from Counter
<< input: 1 >>
THEN: [Sound Object] >> Samples >> Play sample
<< Choose the "VIOLIN1.wav" sound, from \Samples\Music\Jingles >>

```

And the dumped wrench...

```

IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object
<< select: LEFT BUTTON, SINGLE CLICK >>
<< choose the [object.wrench] object >>
THEN: [object.wrench] >> Destroy
THEN: [Create New Objects] >> Create Object
<< Select the [scratched.out] object >>
<< Set the coordinates to x=698, y=37 >>
THEN: [Create New Objects] >> Create Object
<< Select the [white.blast] object >>
<< Set the coordinates to x=-1, y=-1, relative to [object.wrench] object >>
THEN: [counter.objects] >> Subtract from Counter
<< input: 1 >>

```

THEN: [Sound Object] >> Samples >> *Play sample*  
<< Choose the “VIOLIN2.wav” sound, from \Samples\Music\Jingles >>

We’re almost done with this series of events... Here’s one for the globe-shaped ball:













IF: [Keyboard & Mouse Object] >> The Mouse >> *User clicks on an object*  
<< select: LEFT BUTTON, SINGLE CLICK >>  
<< choose the [object.earth] object >>  
THEN: [object.earth] >> *Destroy*  
THEN: [Create New Objects] >> *Create Object*  
<< Select the [scratched.out] object >>  
<< Set the coordinates to x=674, y=36 >>  
THEN: [Create New Objects] >> *Create Object*  
<< Select the [white.blast] object >>  
<< Set the coordinates to x=-3, y=1, relative to [object.earth] object >>  
THEN: [counter.objects] >> *Subtract from Counter*  
<< input: 1 >>  
THEN: [Sound Object] >> Samples >> *Play sample*  
<< Choose the “VIOLIN3.wav” sound, from \Samples\Music\Jingles >>

And here’s the last one, concerning the pile of coins...

IF: [Keyboard & Mouse Object] >> The Mouse >> *User clicks on an object*  
<< select: LEFT BUTTON, SINGLE CLICK >>  
<< choose the [object.coins] object >>  
THEN: [object.coins] >> *Destroy*  
THEN: [Create New Objects] >> *Create Object*  
<< Select the [scratched.out] object >>  
<< Set the coordinates to x=650, y=36 >>  
THEN: [Create New Objects] >> *Create Object*  
<< Select the [white.blast] object >>  
<< Set the coordinates to x=-1, y=-3, relative to [object.coins] object >>  
THEN: [counter.objects] >> *Subtract from Counter*  
<< input: 1 >>  
THEN: [Sound Object] >> Samples >> *Play sample*  
<< Choose the “VIOLIN4.wav” sound, from \Samples\Music\Jingles >>

Got it? Great! That means we’re done with our pickupable items. All we need now are seven more events and we’ll have our game up and running!

Here's what we've got so far (don't panic if isn't exactly the same as in your Event Editor):

All the events All the objects																					
1	• Start of Frame		✓																		
2	• User clicks with left button		✓																		
3	• User clicks with left button on 		✓		✓				✓											✓	
4	• User clicks with left button on 		✓		✓				✓											✓	
5	• User clicks with left button on 		✓		✓				✓											✓	
6	• User clicks with left button on 		✓		✓				✓											✓	
7	• User clicks with left button on 		✓		✓				✓											✓	
8	• User clicks with left button on 		✓		✓				✓											✓	
9	• User clicks with left button on 		✓		✓				✓											✓	
10	• User clicks with left button on 		✓		✓				✓											✓	
11	• User clicks with left button on 		✓		✓				✓											✓	
12	• User clicks with left button on 		✓		✓				✓											✓	
13	• User clicks with left button on 		✓		✓				✓											✓	

Looks neat, right? Let's get going then...

4) This little event will be responsible for destroying the “white.blast” object once its animation has finished:

**IF: [white.blast]>> Animation >> Has an animation finished?**

**<< select “Stopped” >>**

**THEN: [white.blast] > Destroy**

5) Time for our “Always”-conditioned event, one that happens on every loop, no matter what. This event is actually all about positioning – it repositions the “Lens” and “magnifying.glass” objects so that they always follow the player's cursor, and then brings them to front, so that they won't be covered by the white explosion we create in earlier events:

**IF: [Special Object] > Always**

**THEN: [Lens] > Position > Set X position**

**<< input: xmouse >>**

**THEN: [Lens] > Position > Set Y position**

**<< input: ymouse >>**

**THEN: [magnifying.glass] > Position > Set X position**

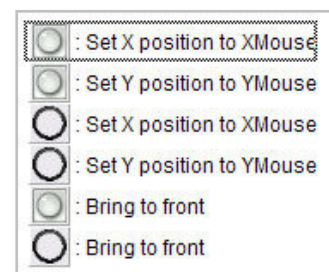
**<< input: xmouse >>**

**THEN: [magnifying.glass] > Position > Set Y position**

**<< input: ymouse >>**

**THEN: [Lens] >> Order >> Bring to front**

**THEN: [magnifying.glass] >> Order >> Bring to front**



6) Here's a simple, yet crucial event – the one controlling our timer...

```
IF: [The Timer Object] >> Every
<< Set the timer to 1 second >>
IF: [counter.objects] >> Compare the counter to a value
<< compare whether it is Different than 0 >>
THEN: [counter.time] >> Subtract from Counter
<< input: 1 >>
```

7) Out of time? Sorry, bub, that means you've officially lost...

```
IF: [counter.time] >> Compare the counter to a value
<< compare whether it is Equal 0 >>
THEN: [Storyboard Controls] >> End the application
```

8) But if you've gathered all the required items, you're gonna' hear a nice little victory sound...

```
IF: [counter.objects] >> Compare the counter to a value
<< compare whether it is Equal 0 >>
IF: [Special Object] >> Limit conditions>> Run this event once
THEN: [Sound Object] >> Samples >> Play sample
<< Choose the "MUSPIAN5.wav" sound, from \Samples\Music\Jingles >>
```

9) ...and then you'll have no more than three seconds to say goodbye to this level...

```
IF: [counter.objects] >> Compare the counter to a value
<< compare whether it is Equal 0 >>
IF: [The Timer Object] >> Every
<< Set the timer to 1 second >>
THEN: [counter.hidden] >> Subtract from Counter
<< input: 1 >>
```

10) ...until you move on to the next one. Of course, if there will be a next one...

```
IF: [counter.hidden] >> Compare the counter to a value
<< compare whether it is Equal 0 >>
THEN: [Storyboard Controls] >> Next frame
```

You know what? That was our last event...

## That's it! Congratulations!

Yes, you've done it, detective! You've solved the case of this tutorial... But there is still a great mystery lingering around the corner: the one with all the eagles, with Joshua Paris and his business partners, with worried Rebecca waiting for any news about her missing brother... The one in which Johnny MacCullagan may need your help. Think about it and try to build new levels, finish this story on your own, helping Johnny to uncover an evil scheme of epic proportions. It's all up to you, Fusioner... It's all up to you.

*Thanks for your time and see you again soon!*

***Cheers!***

**Koobare**

*If you have any questions, suggestions or just need help –  
mail me at [marchewkowy@gmail.com](mailto:marchewkowy@gmail.com)*

**To answer a question that I find in my e-mail box ever so often: yes, I can help you with your game, create you some graphics, design your levels, draw you a menu screen and write you a story... But I cannot do it for free – I actually upkeep myself from my designing, so there's no chance for that. But if you've got some spare cash, you can hire me to create whatever you'd like, you'll receive vehicles, monsters, explosions, dialogues and whatever else you need. **I'm a designer mercenary and I'm always for hire.****

**All copyrighted materials, names, titles, images and visualizations (as "Max Payne" etc.) belong to their respective owners and are used here exclusively as a parody or a satirist fan tribute – no copyright infringement intended.**



You have been reading...

# *the* **MYSTERY** *of Paris*



brought to you by  
**Koobare**

Clickteam's funkiest  
~~mercenary~~  
*Private Eye*



Created for Multimedia Fusion 2 & Multimedia Fusion 2: Developer

Always be sure to have your MMF2 up-to-date!